Rails Revealed
Lenz - Chapter 4

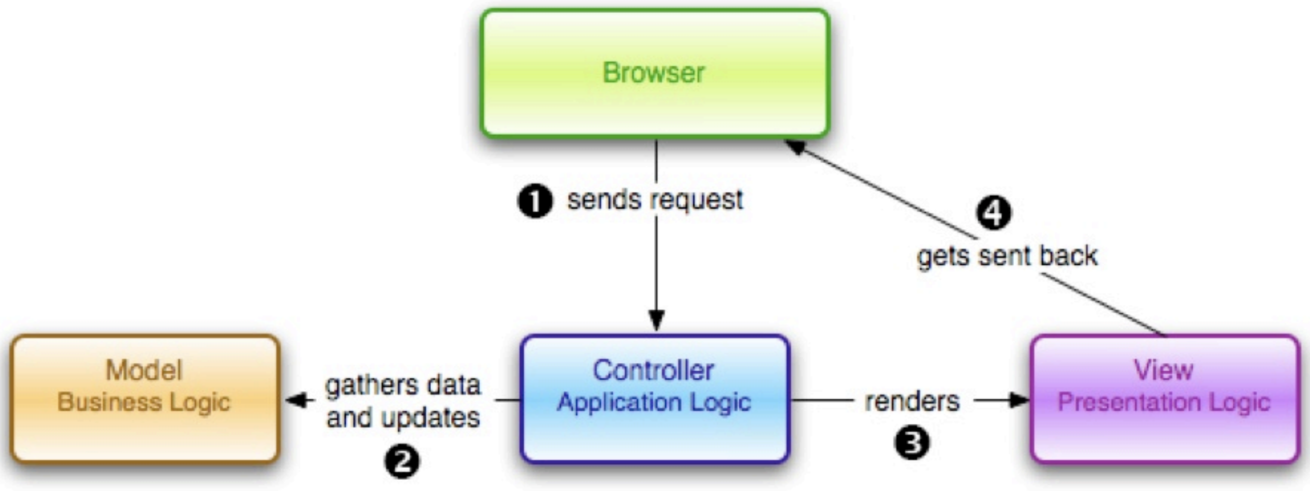
Charles Severance

Textbook: Build Your own Ruby on Rails Application by Patrick Lenz (ISBN:978-0-975-8419-5-2)

Rails Overview

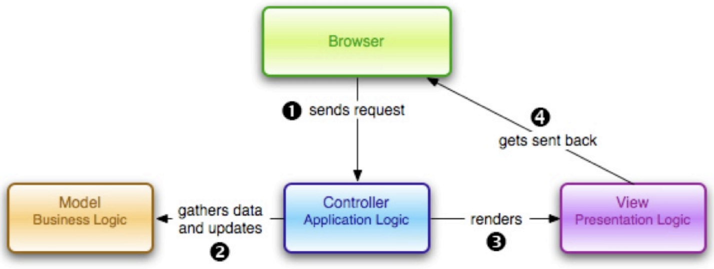
- Rails is a full-stack MVC Framework for web applications
- Rails was built by taking the common elements of a successful complex web application ad generalizing them.
- Rails specifies many aspects of the application development environment so that application developers can focus on the functionality of their application

MVC - Request - Response Cycle



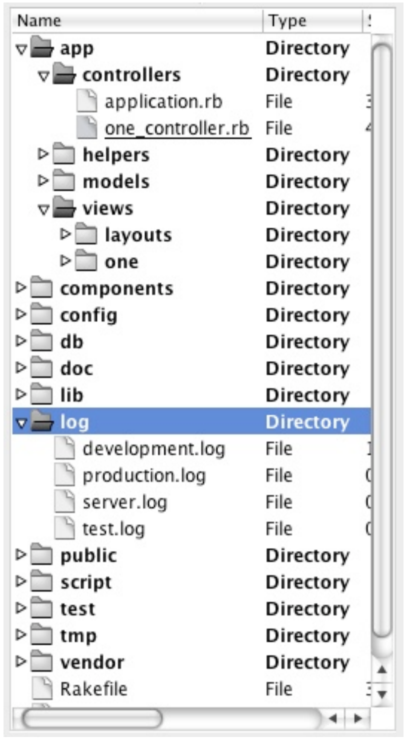
MVC Sequence

- User presses button, browser sends data to application
- Controller receives the data, and makes updates to and/or retrieves from the model as necessary
- User output data is passed to the View - view applies final look and feel and the response goes back to the Brower.



A Rails Application

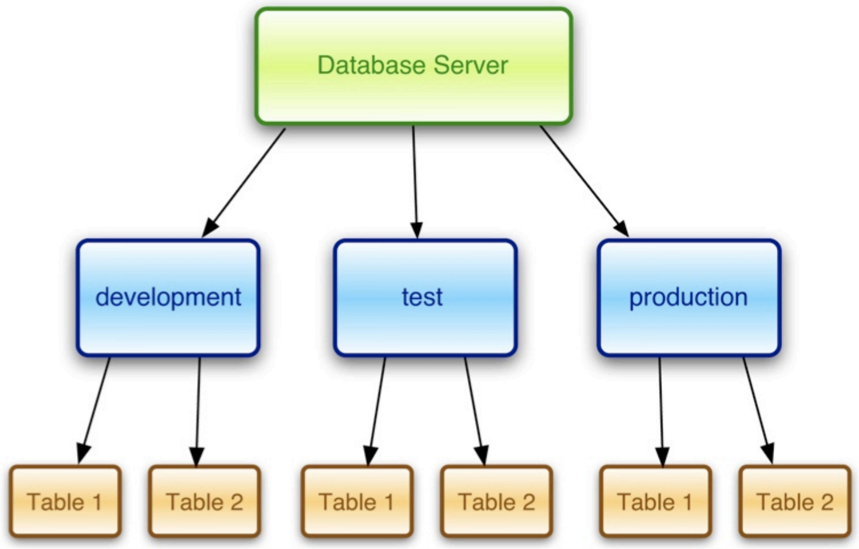
- Rails even dictates the layout of an application directory - one less decision for a developer to make
- The directory structure precisely reflects the MVC architecture
- This helps Rails developers know where to look for things when faced with a new Rails application
- Removing choice improves clarity



```
Name                      Type
▽ app                     Directory
  ▽ controllers           Directory
      application.rb      File
      one_controller.rb   File
  ▷ helpers               Directory
  ▷ models                Directory
  ▽ views                 Directory
    ▷ layouts             Directory
    ▷ one                 Directory
▷ components              Directory
▷ config                  Directory
▷ db                      Directory
▷ doc                     Directory
▷ lib                     Directory
▽ log                     Directory
    development.log       File
    production.log        File
    server.log            File
    test.log              File
▷ public                  Directory
▷ script                  Directory
▷ test                    Directory
▷ tmp                     Directory
▷ vendor                  Directory
  Rakefile                File
```

Database Structure

- Rails supports three database configurations for an application
  - Development
  - Test
  - Production



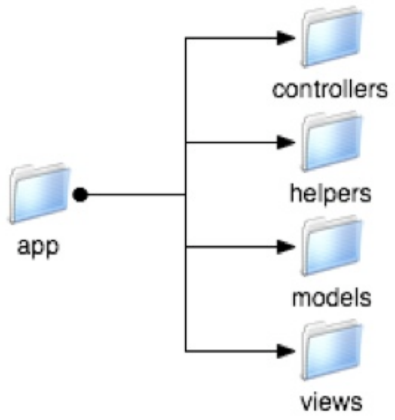
Configuration: config/database.yml

- This mirrors a common pattern used by many mature applications.
- Production - kept separate
- Test - created fresh for each test run
- Development - Allowed to grow between runs - separate from production

```
development:
  adapter: sqlite3
  database: db/development.sqlite3
  timeout: 5000
test:
  adapter: sqlite3
  database: db/test.sqlite3
  timeout: 5000
production:
  adapter: sqlite3
  database: db/production.sqlite3
  timeout: 5000
```

MVC The Rails Way

- The application directory layout directly mirrors the MVC model



The Essential Classes of Rails

- ActiveRecord - Each model object extends this class. The Object-to-Relational Database Mapping (aks ORM) is handled by this class.
- ActionController - Handles browser requests - makes calls to the model to retrieve / update data and prepares data for the view as needed
- ActionView - Takes data from the controller and presents it to the user after properly rendering it.

ActiveRecord

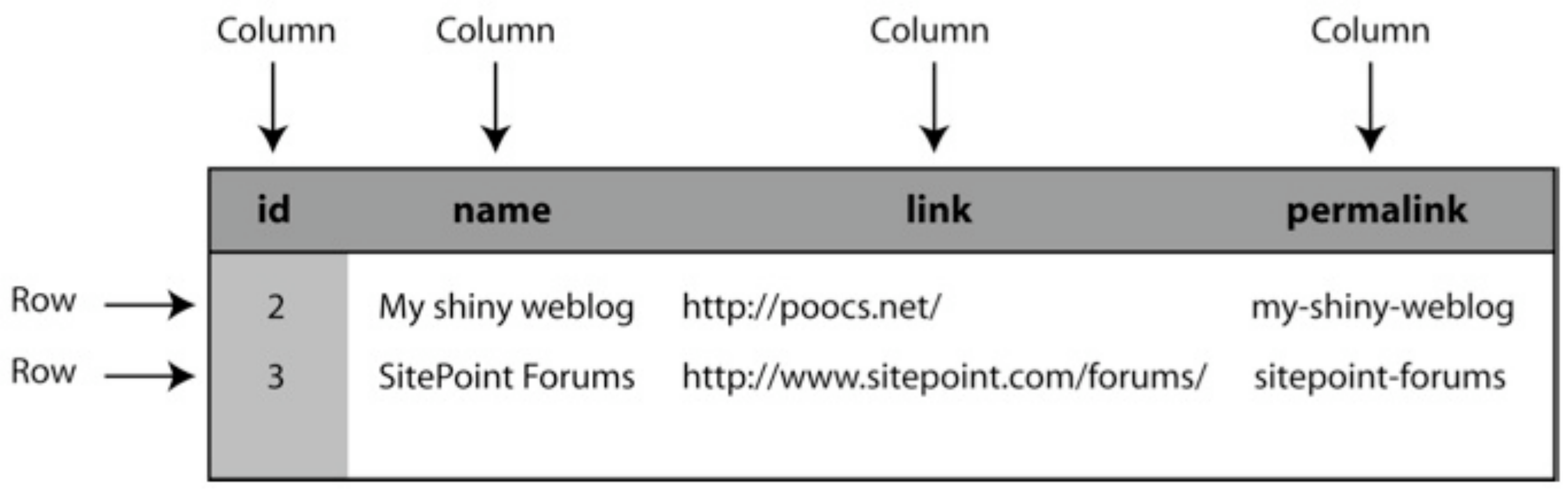
- Connects to the Database
- Retrieves data from tables and makes objects
- Stores new objects in tables
- Provides abstraction layer to insulate Models from different database dialects such as MySql, Oracle, SQLServer, Postgress, etc.
- Keeps Rails applications portable across databases.

http://api.rubyonrails.org/classes/ActiveRecord/Base.html

What is a Database

- Databases are made up of tables - in a way like Excel Spreadsheets
  - Each row is an object
  - Each column in a table has a name and type
- We communicate with databases using a language called Structured Query Language (SQL)
- SQL is a standard but there are many variants of SQL.

Database Table



id	name	link	permalink
2	My shiny weblog	http://poocs.net/	my-shiny-weblog
3	SitePoint Forums	http://www.sitepoint.com/forums/	sitepoint-forums

Column
Column
Column
Column
Row
Row

Two lines of code

```
class Story < ActiveRecord::Base; end
s = Story.find(12)
```

- Look for a Database table named “stories”
- Make sure there is a column called “id” which is an integer key
- Select the row with id = 12
- For each column of data, store the data in the new Story instance, and make a method to set and get the data element
- Plus a whole series of methods to store and retrieve the object to and form the database.

Scaffolding to make Models

- ruby script/generate model Story

Relational Databases

- Databases that are very good at representing and looking up "relationships" between data elements in different tables linked by common values.
  - one-to-one associations
  - one-to-many associations
  - many-to-many associations
- The relational operation which deals with these relations is called "join"

ActionController

```
class StoryController < ActionController::Base
  def index
  end
  def show
  end
end
```

- Handles incoming requests
- Updates the model
- Reads from the model
- Selects and prepares for the View

http://api.rubyonrails.org/classes/ActionController/Base.html

ActionView

- Presentation templating only - should not do any processing or touch the model directly - this requires discipline
- Common view approaches
  - index.rhtml - Embedded Ruby interspersed with HTML
  - data.rxml - Embedded Ruby interspersed with xml
  - code.rjs - Ruby interspersed with Javascript

Embedded Ruby (ERb) Syntax

- <% ruby code %> run this Ruby code
- <%= ruby code %> run this Ruby code and print the return value
- Example
  - Here is your prize <%= @prizemessage %> - congratulations!

Passing Data Between Controller and a View

- The View has access to all of the Controller’s instance variables (variables with a prefix of “@”)

```
class StoryController < ActionController::Base
  def index
    @variable = 'Value being passed to a view'
    x = ‘The view cannot see this variable’
  end
end


<h1>Hello and Welcome</h2>
Here is your reward: <%= @variable %>
```

Summary

- Ruby uses and enforces the Model - View - Controller design pattern for web applications
- The essential capabilities of Rails are accessed through three classes which we extend and make use of
  - ActiveRecord
  - ActionController
  - Action View (generally used via a template pattern)